EV369764791

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

SYSTEMS AND METHODS FOR EDITING XML DOCUMENTS

Inventors:

Jonathan E. Rivers-Moore

and

Eugene N. Veselov

ATTORNEY'S DOCKET NO. MS1-1864US

CROSS REFERENCE TO RELATED PATENT APPLICATION

United States Patent Application having serial no. 10/395,505, a filing date of 3/24/2003, and an attorney's docket number of MS1-1343US, for SYSTEM AND METHOD FOR DESIGNING ELECTRONIC FORMS AND HIERARCHICAL SCHEMAS of Paoli, et al is related to this application.

TECHNICAL FIELD

This invention relates to systems and methods for editing an eXtensible Markup Language (XML) document.

BACKGROUND

XML is increasingly becoming the preferred format for transferring information. XML is a tag-based hierarchical language that is extremely rich in terms of the information that it can be used to represent. For example, XML can be used to represent information spanning the spectrum from semi-structured information (such as one would find in a word processing document) to highly structured information (such as that which is contained in a table). XML is well-suited for many types of communication including business-to-business and client-to-server communication. For more information on XML, XSLT, and XML Schema, the reader is referred to the following documents which are the work of, and available from the W3C (World Wide Web consortium): XML 1.0 second edition specification; XSL Transformations (XSLT) Version 1.0; XML Schema section 1: Structures; and XML Schema section 2: Datatypes.

Before information can be transferred, however, it must first be collected. Electronic forms are commonly used to aid in collecting information into an XML

document. Electronic forms can be governed by a template, which can provide rules by which an XML document can be presented as an electronic form, such as with data-entry fields for entry of data.

To create these templates, however, a programmer often needs a significant understanding of HTML and XML Schemas. A programmer often needs to understand how data-entry fields in an electronic form governed by the template are represented in the schema, HTML file, and XML document. The programmer also may need to understand how HTML, XML, and XML Schemas are structured and how they interrelate. Thus, to build a template, a programmer often must have significant experience and skill.

In addition, to use the template, a programmer may need to build a program to allow a user to edit an XML document governed by this template.

For these reasons, creating and using templates can be difficult, time consuming, and require a programmer of significant skill.

SUMMARY

In the following description and figures, an editing application is described that is capable of identifying nodes of an XML document that are editable and operations permitted for those nodes using elements of an electronic-form template.

In one implementation, the editing application presents an XML document as an electronic form having data-entry fields representing nodes of the XML document. The editing application presents those nodes that are identified as editable in a parent element in another XML document governing the first XML document. The editing application enables certain operations for those editable

nodes through the data-entry fields if a child element of the parent element identifies them.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a system with a display screen, computer, and user-input devices. The system implements a method for designing an electronic-form template. The system also implements a method for editing XML documents using an electronic-form template.

Fig. 2 illustrates an exemplary screen display showing a hierarchical view and a design view of an electronic-form template.

Fig. 3 illustrates an exemplary component display area.

Fig. 4 is a flow diagram of an exemplary process for generating electronic-form templates.

Fig. 5 illustrates an exemplary screen display showing a component display area and a blank form-design area.

Fig. 6 illustrates an exemplary screen display showing a hierarchical view and a design view of an electronic-form template, and a component display area.

Fig. 7 illustrates the exemplary screen display of Fig. 6 after the electronic-form template comprises another component.

Fig. 8 illustrates the exemplary screen display of Fig. 7 after the electronic-form template comprises other components.

Fig. 9 illustrates the exemplary screen display of Fig. 8 showing also a component context menu and a structure submenu.

Fig. 10 illustrates an exemplary hierarchical view display area, a change inquiry window, and an add window.

Fig. 11 is a flow diagram of an exemplary process for editing an XML document using an electronic-form template.

Fig. 12 illustrates an exemplary screen display showing an electronic-form representation of an XML document having one editable node.

Fig. 13 illustrates an exemplary screen display showing an electronic-form representation of an XML document following an electronic-form template.

Fig. 14 is a block diagram of a computer system that is capable of supporting processes for creation and use of an electronic-form template.

The same numbers are used throughout the disclosure and figures to reference like components and features.

DETAILED DESCRIPTION

The following disclosure describes a user-friendly way to design electronic-form templates using components and a form-designing area of a display. Components are presented in an area of a display screen, usually graphically, such as with an arrangement of icons. Icons representing each component are a simplification so that a designer can more easily understand the purpose of and choose from a list of components. A designer can choose each component that he or she wishes to include in an electronic-form template.

The designer can choose a component, such as by clicking on an icon representing a component, and placing it in a form-designing area. The form-designing area is presented in an area of a display screen, usually appearing as a blank page, such as is often done when viewing a new document in a word-processing application. Components placed in a form-designing area can be manipulated by a designer to allow the designer to create an electronic-form

template that provides a particular look and feel for an electronic-form representation of an XML document. Also by choosing particular components, a designer can build into the electronic-form template various types of permitted operations (e.g., editing operations).

With each new component added or modified, and in some cases each change made to an electronic-form design view or its hierarchical view, the electronic-form template (and its views) is altered to reflect that change. This incremental building of an electronic-form template and its views, and the fact that the views are linked so that a change to one can almost instantly be reflected in the other, allows a designer to quickly, easily, and intuitively create electronic-form templates.

The resulting electronic-form template reflects a designer's chosen look, feel, and editing options for nodes of XML documents that are to be governed by the electronic-form template. Thus, by following this electronic-form template, nodes of an XML document can be located within the XML document, displayed, and made editable. In one embodiment, an editor application is used to determine, based on this electronic-form template, how nodes of an XML document are to be edited.

For discussion purposes, the visual representation of the components, hierarchical view, electronic-form design view, and XML document are described in the context of a single computer, a set of user-input devices, and a single display screen having areas for displaying a representation of the components, the electronic-form design view, the hierarchical view, and the XML document. The display screen, computer, and user-input devices will be described first, followed by a discussion of the techniques in which these and other devices can be used.

System Architecture

Fig. 1 shows an exemplary implementation of various devices and applications that can be used to facilitate the creation of an electronic-form template from components and enable editing of XML documents governed by the created electronic-form template.

Fig. 1 shows an exemplary system 100, which comprises a screen 102, user-input devices 104, and a computer 106.

The user-input devices 104 can comprise any device allowing a computer to receive a designer's preferences (or edits from an end-user), such as a keyboard 114, other device(s) 116, and a mouse 118. The other input devices 116 can comprise a touch screen, a voice-activated input device, a track ball, and any other device that allows the system 100 to receive input. The computer 106 comprises a processing unit 120 and random access memory and/or read-only memory 122 including applications, such as an operating system 124, a design application 126, a user interface 128, an editor application 130, an electronic-form template 132, and an XML document 134. The computer 106 communicates with a designer and end-user through the screen 102 and the user-input devices 104.

The screen 102 comprises three displays or screen areas: a hierarchical view display area 108; a component display area 110; and an electronic-form-design area 112. With these areas, a designer can see a representation of and select a component from a list of components. Any part or all of the screen 102 can be used to present an electronic-form representation of the XML document 134. With this representation an end-user can edit or view the XML document 134.

Fig. 2 shows an exemplary design screen 200, including an example of the form-design area 112 and the hierarchical view display area 108 (entitled "Data Source"). Partially within the form-design area 112 is a design view 202 of an exemplary embodiment of the electronic-form template 132. This design view 202 provides information useful to aid a designer in constructing the electronic-form template 132. As will be shown later, an electronic-form representation of the XML document 134 for editing and/or viewing by an end-user (e.g., a data-entry user) can appear similar to this design view 202 of the electronic-form template 132, though this is not necessary. In cases where the electronic-form representation contains data-entry fields and similar user interfaces, these user interfaces can reflect operations permitted by components added to the electronic-form template 132 by the designer.

This electronic-form template 132 shown in the design view 202 is being built from components chosen by a designer. The components chosen are used by the design application 126 to create the data-entry fields shown in the design view 202. These data-entry fields are one way in which the electronic-form template 132 can be represented to a designer or an end-user. These data-entry fields correspond to parts of the electronic-form template 132, the parts also being shown through the icons displayed in the hierarchical view display area 108. The icons displayed are a representation of part of the electronic-form template 132 and are arranged into a tree structure.

Fig. 3 shows an example of components from which a designer can choose, which are displayed here at the component display area 110. These various components comprise a text box 302, a rich text box 304, a drop-down list box 306, a list box 308, a date picker 310, a check box 312, an option button 314, a

section 316, an optional section 318, a repeating section 320, a repeating table 322, a bulleted list 324, a numbered list 326, a plain list 328, a button 330, and hyperlink 332. Other components can be included as well. As described in further detail below, each of these components can be added to indicate an operation or operations that is permitted to be performed on a node or nodes of the XML document 134. By way of example, some of these components will be represented in an electronic form by data-entry fields enabling the component's operation. These data-entry fields can be associated with appropriate nodes of the XML document 134. Thus, when an end-user uses an electronic form's data-entry field to make an entry, for instance, the entry can be reflected in the XML document 134.

With the listed components and other components the system 100 enables a designer to build the electronic-form template 132. These components enable many different possible types of operations (and user interfaces), such as those shown with various data-entry fields in the design view 202 in the form-design area 112 of Fig. 2. The process used to build the electronic-form template 132 will be set forth in greater detail below.

The above devices and applications are merely representative, and other known devices and applications may be substituted for or added to those shown in Fig. 1. One example of another known device that can be substituted for those shown in Fig. 1 is the device shown in Fig. 14. Other examples include portable, handheld, or wireless devices.

Techniques for Building Electronic-Form Templates

Overview

A system, such as the system 100 of Fig. 1, displays components to a designer. The designer can choose from the components to graphically and easily build the electronic-form template 132. The system 100 can also incrementally build the electronic-form template 132 with each new component the designer adds. The system 100 also allows the designer to see two views of the electronic-form template 132, here the design view 202 and the hierarchical view 204. The system 100 then alters each view with each new component chosen. The designer may also change components existing in the electronic-form template 132, the change to each being incrementally reflected in the views by the system 100.

Fig. 4 shows a process 400 for generating the electronic-form template 132. The process 400 and the following processes are illustrated as a series of blocks representing individual operations or acts performed by the system 100. These processes may be implemented in any suitable hardware, software, firmware, or combination thereof. In the case of software and firmware, the processes represent a set of operations implemented as computer-executable instructions stored in the memory 122 and executable by the processing unit 120.

Displaying Components and Form-Design Area

At block 402, the user interface 128 displays components and a form-design area. It does so to enable a designer to graphically design the electronic-form template 132.

Fig. 5 shows a design screen 500 created by the user interface 128, having an example of the component display area 110 and a blank example of the form-

design area 112. The form-design area 112 is displayed to make it easy for a designer without typical programming skills to create the electronic-form template 132.

To make it easy, the user interface 128 can provide an editing experience to a designer similar to that commonly provided in word-processing systems. The user interface 128 can, for instance, work like a word-processing system by providing similar font controls and options. In Fig. 5, for example, the user interface 128 displays the form-design area 112 looking like a page from a word-processing application—here, a blank white page. It can also display commonly used icons that represent operations that a designer can choose to perform, such as the font being used (in Fig. 5, Verdana, size 10), bold/underline/italic options, and the like. These word-processing icons can be displayed in many different ways, including as shown in a word-processing icon display 502 of Fig. 5.

Also, as stated elsewhere herein, changes made by the designer to the form-design area 112 can be reflected in the form-design area 112 instantaneously (from the perspective of the designer), further making the design process similar to a typical word-processing experience. By so doing, the user interface 128 makes designing the electronic-form template 132 simpler and more intuitive for a person skilled in word-processing.

The components are displayed by the user interface 128 in the component display area 110 to make it easy for a designer without extensive knowledge of components to be able to understand what each of them can represent in the electronic-form template 132. To show what each component represents, the user interface 128 displays icons and/or text to inform the designer, such as with the icons and text set forth in the component display area 110 set forth in Figs. 3 and

5. In Fig. 3, for example, the text box 302 comprises an icon (i.e., a symbol) and text describing what a text box component represents. This icon shows a designer that, should he choose to include a text box component in his electronic-form template 132, an operation allowing entry and editing of text for a node in the XML document 134 will be added to the electronic-form template 132. The editor application 130 can then present the node of the XML document 134 in an electronic form as a data-entry field allowing input of text. Like the icon, the text describing the text box 302 ("Text Box") is also descriptive.

With the component display area 110 and the form-design area 112 displayed, the designer can begin to build the electronic-form template 132 and view what an electronic-form representation of the XML document 134 can look like. He can continue to build the electronic-form template 132 by adding components, but can also alter the electronic-form template 132 by altering existing components. This process of building and altering is shown as a design sub-process 403, which includes blocks 404 to 412. The sub-process 403 includes blocks used to describe the action and interaction of the designer and the system 100. When the designer has finished with the electronic-form template 132, the design application 126 produces an electronic-form representation mirroring the operations allowed by the electronic-form template 132 (block 414). The process 403 and the block 414 will be described in greater detail below.

When the component display area 110 and the form-design area 112 are presented, the designer can pick a component from the list of components in the component display area 110 for insertion into the form-design area 112 (block 404). The designer can pick from components in various ways, including through the mouse 118, the other devices 116 (such as a touch screen, track ball, voice-

activation, and the like), and through the keyboard 114, which are shown in Fig. 1. To grant flexibility to the designer, the system 100 enables the designer to move the component in the form-design area 112 to where she desires.

A designer can pick a component, for example, by dragging and dropping (from the component display area 110) a component's icon onto a form-design area 112 shown in Fig. 5. The designer can pick a component to drag and drop with various devices, such as with the mouse 118 or commands entered through the keyboard 114. In Fig. 5, the designer clicks on the icon and text for the text box 302 to select it.

How an icon for a component looks may not exactly indicate how it will look in an electronic-form representation of the XML document 134 that follows the electronic-form template 132. Icons, for instance, are often too small to be exact. Rather, icons are designed to indicate functions or operations (e.g., editing functions) associated with nodes of the XML document 134 that choosing the component will permit.

Fig. 6 shows an exemplary screen display 600 showing what the design application 126 creates after a designer selects the text box 302 in Fig. 5 (also shown in Fig. 6). In this example, the system 100 creates a text-function node 604 allowing entry of text, here represented by the text-box data-entry field 602, which looks like a gray box for entry of text and is labeled "String 1:". The design application 126 enables the designer to continue building his electronic-form template 132 by selecting components, thereby creating certain allowed operations associated with nodes of the XML document 134. In this example, the design application 126 created part of the electronic-form template 132 that indicates a permitted function (here a text-entry function) with the text-function node 604.

Building an Electronic-Form Template

Once the system 100 receives a selection of a component and the placement for the component, the system 100 can identify which component was selected, identify the placement for that component on the form-design area 112, build the electronic-form template 132 based on the component chosen and its location, and display the design view 202 and the hierarchical view 204. These tasks are set forth in blocks 406, 408, 410, and 414 of Fig. 4, which will be described below.

In block 406, the design application 126 identifies which component was selected. The system 100 can access and/or contain many components, either from local or remote sources. Some of these components are set forth (via icons and text) in the component display area 110 shown in Figs. 3, 5, 6, and 7.

Also in the block 406, the design application 126 identifies where a component is placed in the form-design area 112. The placement of the component can alter the structure of the electronic-form template 132. How the placement of a component can alter the electronic-form template 132 will be set forth in greater detail below.

If, for example, a designer chooses the text box 302 from the component display area 110 of Fig. 5, and places the text box 302 in the upper left corner of the form-design area 112, the design application 126 will identify the component and this placement. With this information, the system 100 proceeds to build the electronic-form template 132, which will be described in part using Figs. 5 and 6.

In block 408, the design application 126 changes the electronic-form template 132 based on a component selected. When a component is added, as has been described in part above, the design application 126 also changes the

hierarchical view 204 and the design view 202 of the electronic-form template 132 by building in a representation of the added component. When an existing component is altered (discussed in greater detail below), the design application 126 changes the views to reflect that alteration.

Generally, when a component is added to the form-design area 112, one or more operations are then permitted by the electronic-form template 132. A syntax (also called a "character string") corresponding to each of these permitted operations can be added by the design application 126 to the electronic-form template 132. An exemplary list of operations and their related, operational syntaxes are shown below:

Component Chosen	Operation Permitted	Syntax Added
Repeating Section Repeating Table	Insertion of Nodes Before or After Associated Node, and Removal of Nodes	xCollection
Optional Section	Insert or Remove Nodes	xOptional
Plain List Bulleted List Numbered List	Text Editing of Data Within Associated Node(s), Plus Merge, Split, and Removal of Subordinate Node(s)	xTextList
Rich Text Box Text Box	Text Editing of Data Within Associated Node(s)	xField
Picture	Insertion of Images Into Associated Node(s)	xImage

An operation permitted can govern, for instance, how and what kind of information is permitted to be added into a node or nodes of the XML document 134 that is associated with that operational syntax. The node or nodes of the XML document 134 can be associated with the operational syntax. The operational syntax can be associated with a location syntax, such as an XPath expression.

Both the operational syntax and the location syntax can be included within an XML element added to the electronic-form template 132. XML elements will be described in greater detail below.

Fig. 7 shows an exemplary screen display 700 showing the continued building of the electronic-form template 132. Here, the designer chose another component (the check box 312 of Fig. 6) to add to the electronic-form template 132 shown in Fig. 6.

In one embodiment, when the designer adds the Rich Text Box component 304 to the form-design area 112, the design application 126 adds the following XML element to the electronic-form template 132:

```
<xsf:xmlToEdit name="xhtml1" item="/myFields/xhtml1" >
  <xsf:editWith type="rich" component="xField" />
</xsf:xmlToEdit>
```

This XML element is an example of an "editability element", described below.

In response to the designer's choice, the design application 126 represents the added editability element as a Rich Text Box 702. As shown in the screen display 700, the Rich Text Box 702 is labeled "Xhtml 1". Also in response to the designer's choice, the design application 126 altered the hierarchical view 204 to include an XHTML section 704 (labeled "xhtml1") corresponding to the Rich Text Box 702 and the editability element.

This editability element comprises XML attributes and additional child elements. These elements and attributes provide information that the editor application 130 can use to determine that a node or nodes of the XML document

134 are editable. These elements and attributes also provide the editor application 130 with information indicating which operations are permitted for these editable nodes.

Syntax of the editability element indicates a name and/or location in the XML document 134 of a node or nodes. With this indicator, the editor application 130 can determine which node or nodes are potentially editable with operations permitted by this editability element. In one implementation, by using all of the electronic-form template 132, the editor application 130 can determine all nodes in the XML document 134 that are potentially editable.

Continuing the ongoing example, the "xmlToEdit" editability element includes an attribute having a character string of "item", associated with an indicator of: "myFields/xhtml1" (the "value" of the attribute). This indicator can be used to determine that nodes having the name and/or location of "myFields/xhtml1" are potentially editable. In this example the indicator comprises an XPath expression being usable to locate node(s) matching this XPath expression.

An additional attribute (a 'container' attribute) can also be included, to indicate contextual conditions, if any, that should be present for the node to be deemed editable. In the ongoing example, there is no 'container' attribute. Thus, in this example, the node indicated by "myFields/xhtml1" is editable without contextual conditions.

Another child XML element, called an "operation element", in the editability element indicates operations (e.g., editing functions) permitted to be performed on or with the node(s) indicated in the editability element. In the ongoing example, the operation element entitled "editWith" indicates that

operations associated with "xField" of type "rich" may be applied on or for the nodes indicated by "myFields/xhtml1".

The operation or set of operations indicated by the operation element can be determined from the syntax given in the operation element. Thus, in the ongoing example the syntax "xField" with the additional "type" attribute, with value "rich", indicates an operation of rich-text editing, such as deleting and inserting text, changing the formatting of text, making it bold or italics, and other familiar rich-text editing operations familiar to word-processing. In this example, then, the nodes or data within the nodes indicated by "myFields/xhtml1" may be edited using these operations.

The operation element can indicate permitted operations in various manners. In one implementation, a syntax of the operation element includes an element specifying the XML data able to be inserted by the associated operations.

Thus, the electronic-form template 132 governs how information is handled and operations permitted for the XML document 134 for which it corresponds. Because the electronic-form template 132 comprises an editability element and its parts for each component chosen by the designer, the chosen component affects the syntax of the electronic-form template 132.

With the above syntax and the editability element, each component built into the electronic-form template 132 can govern the look, display, orientation, and size of data-entry field(s), as well as how and where information entered into these data-entry field(s) is mapped, such as to nodes of the XML document 134. The editor application 130 can, however, use less than all of the electronic-form template 132. It can, for instance, determine potentially editable nodes and

operations permitted, but use a different look, display, orientation, and size of data-entry fields (or not use data-entry fields at all).

Once the system 100 changes the electronic-form template 132 (block 408), it proceeds to block 410 to display the electronic-form template's 132 design view 202 and hierarchical view 204, or to block 414, to complete the process and produce the electronic-form template 132. Block 410 will be discussed first below, followed later by a discussion of block 414.

In the block 410, the user interface 128 displays the electronic-form template's 132 views. The hierarchical view 204 can be represented in various ways, including by visually showing the hierarchical structure of the electronic-form template 132, if applicable. One example of this is the indentations of the hierarchical view 204 set forth in the hierarchical view display area 108 of Fig. 2.

The electronic-form template 132 can be represented in various ways. The user interface 128 can display the design view 202 to make it easy for a designer to understand the structure of the electronic-form template 132, or can show the electronic-form template 132 without additional information. The design view 202 of the electronic-form template 132 can mimic an electronic-form representation of the XML document 134 potentially seen by a user, such as set forth in the form-design area 112 of Fig. 2. The design view 202 can show components built into the electronic-form template 132 with additional information showing details about the data-entry field or fields corresponding to the components to aid a designer in understanding and altering the components in the electronic-form template 132.

The user interface 128 can also display the hierarchical view 204 in order for the designer to assess the electronic-form template 132 after each change made

by the designer. When the designer can quickly see the effect of his changes, the designer can more easily build the electronic-form template 132 to match the designer's intended results. Once the design view 202 is displayed and the hierarchical view 204 is displayed (if desired) in the hierarchical view display area 108, the designer can continue to make changes or end the process. Ending the process will be discussed as part of the block 414, described below.

Continuing the example above, the design application 126 will build the electronic-form template 132 based on the identity and placement of the component.

One of the advantages of the design application 126, and the method it employs, is that the electronic-form template 132 can be built incrementally. That is to say, with each component chosen, or in some implementations with each other action taken that will affect a view of the electronic-form template 132, the electronic-form template 132 is altered. This incrementalism allows a designer to quickly see how the electronic-form template 132 is changing with each change made by the designer. The designer does not have to work on either a form or a schema and then save the form or schema to see how the corresponding schema or form looks or performs. Instead, as the designer makes a change, it is reflected in the views of the electronic-form template 132. This makes designing an electronic-form template 132 easy and intuitive.

In one implementation, the design application 126 can reflect each change made by a designer to both views of the electronic-form template 132 so quickly that the designer sees it in real time, regardless of whether the change was made to the electronic-form template 132 by altering the design view 202 or the

hierarchical view 204. By so doing, the designer can more easily design electronic-form templates.

With the new change to electronic-form template 132 shown, the design application 126 continues to enable the designer to add components to the electronic-form template 132, returning to block 404, or alter an existing component, block 412.

If the designer chooses to add another component, the design application 126 enables him to do so in a manner similar to picking the first component as described above. The design view 202 Fig. 2 is one view of an electronic-form template built from many components.

Fig. 8 shows an exemplary design screen 800 showing the continued building of the electronic-form template 132. The design screen 800 shows the views of the electronic-form template 132 from Figs. 6 and 7, and the results of the designer continuing to choose components. Through this process of adding components to the form-design area 112, a designer can build everything from a simple electronic-form template, such as shown in Fig. 6, to a moderately complex electronic-form template, such as shown Fig. 8, to a large, complex electronic-form template, such as shown in Fig. 2.

In one embodiment, when a designer adds the repeating section component 320 (shown in Fig. 3 and Fig. 6) to the form-design area 112, the design application 126 adds the following editability element to the electronic-form template 132:

<xsf:xmlToEdit name="repeating_string"

```
        item="/myFields/container2/repeating_string"
        container="/myFields /container2" >
        <xsf:editWith component="xCollection"/>
    </xsf:xmlToEdit>
```

In this example, the editability element (entitled "xmlToEdit") includes an attribute having a character string of "item", associated with an indicator of: "myFields/container2/repeating_string" (the "value" of the attribute). This indicator gives a name/location of nodes in the XML document 134 that are potentially editable. In this example the indicator comprises an XPath expression being usable to locate (i.e., identify) the nodes.

This editability element also comprises a "container" attribute having a value, in this example, of: "myFields/container2" . This attribute determines a dynamic contextual condition for being able to edit the potentially editable nodes indicated by "myFields/container2/repeating_string". In this implementation they are editable only if a node matching the XPath expression: "/myFields/container2" (an ancestor of "myFields/container2/repeating_string") is shown (or rendered) as a section within an electronic-form representation of the XML document 134, and if the user of the editor application has selected a field somewhere within that section. In this example, the condition is met if the data-entry field is rendered within the section corresponding to the "myFields/container2" node's. If not, the editability element does not permit its operations for the potentially editable node(s).

This editability element also contains an operation element entitled "editWith". This operation element indicates that operations associated with

"xCollection" may be applied on or for the nodes indicated by "myFields/container2/repeating_string", if the condition is met.

The operation or set of operations indicated by the operation element can be determined from its syntax. Thus, in the ongoing example the syntax "xCollection" indicates a certain set of operations, described in part above. The set indicated by "xCollection" comprises selecting, inserting, deleting, pasting, and copying the potentially editable nodes (indicated by "myFields/container2/repeating_string") or their subordinate nodes.

In the ongoing example, a repeating section 804 is shown in the design view 202 of the XML document 134. This repeating section 804 comprises data-entry fields shown at numerals 810, 812, and 814. These data-entry fields represent the potentially editable nodes for the editability element.

The editor application 130 can determine that the three nodes matching the XPath expression "myFields/container2/repeating_string" can be edited. Thus, by following the electronic-form template 132, the editor application 130 can correctly enable editing according to the "xCollection" syntax of the operation element. In one implementation, the editor 130 can present the data-entry fields 810, 812, and 814 in an electronic form (e.g., through the user interface 128). The editor application 130 can also enable permitted operations on these data-entry fields (and thus their associated nodes of the XML document 134). In this implementation, the editor application 130 enables selection of and insertion, deletion, pasting, and copying of the editable nodes associated with the data-entry fields 810, 812, and 814.

Returning to the process 400 of Fig. 4, a designer can simply make a change, like adding a component, to a view and see the change applied to both

views. In this sense, the design view 202 and the hierarchical view 204 are actively linked. This active linkage makes designing and changing the electronic-form template 132 quicker, easier, and more intuitive.

In the block 412, the design application 126 enables a designer to select and alter existing components included in the electronic-form template 132. The design application 126 allows the designer to intuitively and easily alter the views of the electronic-form template 132, such as by including editing tools familiar to designers that know word-processing techniques. A designer can change a component stylistically (such as the font, color, size, and the like) and structurally (such as by changing a text box to a check box, and whether or to which other components the component governs or is subordinate). A designer can make these changes also by altering how a component (such as one displayed as a data-entry field) is represented on the design view 202. For example, a designer can click on a component on the form-design screen 112, change the style, move it, delete it, and the like. As the designer makes changes, the design application 126 alters the hierarchical view to continue to correspond to the altering electronic-form template 132 and its design view 202.

Fig. 8 shows the exemplary design screen 800, which provides another example of the design view 202 and the hierarchical view 204 of an example of the electronic-form template 132. To enable the designer to make changes to a component, the design application 126 (through the user interface 128) enables the designer to click on components displayed in the design view 202 of the electronic-form template 132. One such component, a text box data-entry field 802 (labeled "String 5"), is shown as an example. Once the designer selects a component, in this example the text box data-entry field 802, the design

application 126 provides the designer with multiple options to change the data-entry field. As seen in a design screen 900 (described below), the design application 126 provides options in a way comfortable to a designer familiar with common word-processing techniques and icons. If the designer clicked on the text box data-entry field 802 of Fig. 8, the design application 126 can provide multiple pop-up menus of options for the designer.

Fig. 9 sets forth the exemplary design screen 900 including multiple ways in which the design application 126 provides options for a designer. These comprise a component context menu 902 and a structure submenu 904. In this example, the design application 126 enables the designer to change the electronic-form template 132 by changing a representation of a component in the form-design area 112 (such as a data-entry field or accompanying graphics and text). Also, the design application 126 enables the designer to cut the component and move it (through selection and deleting or dragging the component), and change its font, borders, shading, and other style changes (through the component context menu 902), as well as change its structure (through the structure submenu 904). In this example, the designer changes the component by changing the structure of a data-entry field corresponding to the component (the text box data-entry field 802) into a date picker data-entry field by selecting a date picker component 906.

Fig. 10 shows an example of the hierarchical view display area 108, and how it can be used by a designer to alter the electronic-form template 132. In this example, a designer selected an integer node 1002 representing an editability element of the electronic-form template 132. Once chosen, the design application 126 prompts the designer, asking for information, such as through a change inquiry window 1004. Using the window 1004, the design application 126 enables

the designer to make various changes to the editability element represented by the node 1002. He can change, for instance, the data type and default value allowed for this editability element in the electronic-form template 132.

Also as part of this example, the design application 126 presents the designer with the current name, type, and data type of a selected editability element. The designer can then make changes by altering these current parameters of a selected editability element, such as set forth in the change inquiry window 1004 by changing the name to "myNewField" from "element" and the data type to "Text (string)" from "Whole Number (integer)" for the editability element represented by the node 1002.

Fig. 10 shows an example of a pop-up window with which the design application 126 can enable a designer to alter editability elements. With an alter window 1006, the designer can add or delete aspects of an editability element, such as a name and repeatability of an element or establish whether or not it is allowed to be blank.

As part of enabling the designer to makes these changes, the design application 126 makes appropriate changes to the electronic-form template 132 and its views. If the designer deletes a component, for instance, the design application 126 may delete the syntax (e.g., the editability element) corresponding to the component from the electronic-form template 132.

According to block 414, when finished, the end product is the electronic-form template 132. One example of an electronic-form template created by the design application 126 is the purchase order example of the design view 202 of Fig. 2. This purchase order electronic-form template 132 can be used by the editor application 130 to present an electronic-form representation of the XML document

134 with appropriate permitted operations. The editor application 130 can enable this electronic-form representation to be edited by an end user, such as by allowing that user to perform an operation like keying in information into some of the data-entry fields. After entry of information, the information can be stored in the XML document 134.

The information stored can conform to the electronic-form template 132, thereby allowing easy use and/or transfer of information stored in the XML document 134. The electronic-form template 132 can be written in various languages, including schemas written to govern markup languages (such as XML). Schemas governing XML documents are commonly called XML Schemas, DTD (Document Type Definition) schemas, and XDR (XML-Data Reduced) schemas.

The electronic-form template 132 can govern many different documents. Because of this, the electronic-form template 132 can be used to enable thousands of different users keying information into thousands of different documents, all of which can be governed by the electronic-form template 132.

More on Editing XML Documents

As discussed in part above, the electronic-form template 132 comprises information by which an application can determine which nodes of an XML document are editable and in what manner. The editor application 130 of Fig. 1 is described above as one embodiment of an application that can use the electronic-form template 132 to aid in editing XML documents. Here the XML document 134 is assumed associated with the electronic-form template 132. This association can be determined in various manners, such as by reading text in the XML document 134 indicating a relationship with the electronic-form template 132.

Fig. 11 shows a process 1100 for editing the XML document 134. As part of this editing, the editor application 130 uses the electronic-form template 132 to determine which nodes of the XML document 134 are editable. The electronic-form template 132 can also aid the editor application 130 in determining in what way those nodes are editable.

At block 1102, the editor application 130 determines, using the electronic-form template 132, that a node of the XML document 134 is potentially editable. The editor application 130 can do so by reading and analyzing editability elements in the electronic-form template 132.

In one embodiment of the electronic-form template 132, the editability element comprises an indicator with a character string of "xmlToEdit". This string indicates that a node or nodes associated with the indicator are editable (or potentially editable). This editability element also comprises an "item" attribute, which is usable to identify which nodes are editable. An XPath expression identifying the editable nodes is associated with this attribute (the "value" of the attribute). Thus, the editor application 130 can use the XPath expression in the editability element to determine which nodes of the XML document 134 are editable.

For example, if the electronic-form template 132 comprises the following editability element:

```
<xsf:xmlToEdit name="Order"
        item="/Document/Orders/Order" container="/Document" >
        <xsf:editWith component="xCollection"/>
</xsf:xmlToEdit>
```

then the "xmlToEdit" editability element includes an "item" attribute, whose "value" is an XPath expression "/Document/Orders/Order". This XPath expression indicates to the editor application 130 that nodes in the XML document 134 that match this expression are editable (or potentially editable).

At block 1104, with the node or nodes of the XML document 134 identified, the editor application 130 presents data of the editable node. The editor application 130 can present this data using an electronic-form representation of the XML document 134 and/or the editable node. The editor application 130 can do so with aid from the user interface 128 or otherwise.

Fig. 12 shows an example of an electronic-form representation 1200 of the editable node. Here the editor application 130 presents the data of the editable node as an order data-entry field 1202. As shown, a data-entry field presenting data of an editable node can be blank, such as when no information has been entered into the data-entry field. A data-entry field can also include information previously stored in the editable node, which in some cases can be edited, depending on what operations are permitted for the editable node.

At block 1106, the editor application 130 determines what operations are permitted to be performed on the editable node. This determination can be performed by the editor application 130 prior to presenting the data of the editable node. In some cases determining the operations permitted can affect how data is presented. The editor application 130 determines the permitted operations based on an operation element of the electronic-form template 132, discussed in part above. The operation element and the editability element are associated, such as by the operation element being a child element to the editability element. The

operation element indicates operations permitted to be performed on the editable node.

In the ongoing example,

```
<xsf:xmlToEdit name="Order"
        item="/Document/Orders/Order"
        container="/Document" >
        <xsf:editWith component="xCollection"/>
</xsf:xmlToEdit>
```

the operation element comprises a character string of "editWith". This character string indicates to the editor application 130 that the operation element indicates certain types of operations that are permitted. Following this character string, another character sting of "component" is included. This character string can indicate that a value associated with the "component" string that is usable to determine permitted operations.

In the ongoing example, the operation element comprises an operational syntax of "xCollection". This syntax indicates the operations permitted to be performed on nodes of the XML document 134 matching the XPath expression "/Document/Orders/Order".

The editor application 130 can determine the operations associated with this syntax in various ways. The editor application 130 can use this syntax to locate executable code for this operation. In one embodiment of the electronic-form template 132, executable code for operations indicated by various syntaxes are included within the editor application 130.

At block 1108, the editor application 130 enables the permitted operations. The editor application 130 can do so by supplying a user interface appropriate for the permitted operations. For data alteration or addition, the editor application 130 can provide a word-processor-like experience in a data-entry field. Like shown in Fig. 12, the order data-entry field 1202 can be used as a user interface to enter data, delete data, copy and paste data, and the like. Thus, the editor application 130, by determining editable nodes and permitted operations for them, can enable certain types of editing for various editable nodes of the XML document 134.

At block 1110, the editor application 130 receives a selection of an enabled operation. Continuing the ongoing example, the editor application 130 can receive entry of "ACME Tire Company" into the order data-entry field 1202.

At block 1112, the editor application 130 alters data in the editable node (or the editable node itself or associated nodes) of XML document 134 by performing the selected operation. For the ongoing example, the editor application 130 can receive the text "ACME Tire Company" and alter data within the order node to reflect this text. Thus, the editor application 130 can add the text "ACME Tire Company" to the order node.

Other Permitted Operations

The electronic-form template 132 can permit many different types of operations. These comprise, for instance, operations mentioned above and those described below.

The editor application 130 can use the electronic-form template 132 to determine what operations are permitted. With this determination, the editor application 130 can enable permitted operations.

For example, for an editability element of:

```
<xsf:xmlToEdit name="workItem" item="workItems/workItem"
      container="workItems">
    <xsf:editWith component="xCollection" >
      <xsf:fragmentToInsert>
        <xsf:chooseFragment>
          <workItem description="create visuals"></workItem>
        </xsf:chooseFragment>
      </xsf:fragmentToInsert>
    </xsf:editWith>
  </xsf:xmlToEdit>
```

The editor application 130 can determine that the electronic-form template 132, for the node(s) indicated by "workItems/workItem", permits an operation of inserting nodes adjacent to this "workItem" node (or under the node corresponding to the container attribute: "workItems"). In this example, the operation element comprises a syntax of "xCollection", which indicates a permitted operation of inserting or deleting subnodes adjacent to the editable node indicated by "workItems/workItem".

This operation, however, is conditional based on a context element (here an attribute of the editability element). This context is given by a value for a "container" attribute. Here, the container attribute's value is given by a syntax "workItems". This syntax comprises a XPath expression. Thus, the editor application 130 can determine, based on the syntax given, that operations are not

permitted unless a node locatable with the XPath expression "workItems" is exposed. The editor application 130 can determine if the "workItems" node (the parent of the "workItems/workItem" node) is exposed, if a section associated with it is exposed in an electronic-form rendering of the "workItems" node. If the editor application 130 is rendering the XML document 134, the editor application 130 can determine if the "workItems" node is exposed by checking whether it or its user interface has exposed the node.

Additional information useful in performing operations can also be included in the electronic-form template 132. For the example immediately above, syntax can also be included in the operation element to indicate what data is to be inserted in the XML document by the operations. This syntax comprises a "fragmentToInsert" child element to the editability element. Using this information, the editor application 130 can determine that a child element of the fragmentToInsert element contains the data to be inserted.

In this example, a child element having a character string of "chooseFragment" indicates that new XML content corresponding to the XML markup: "<workItem description="create visuals"></workItem> " can be inserted next to the "workItem" node.

In a similar way, the electronic-form template 132 editability element provides syntax permitting deletion of the editable node or nodes using the syntax of "xCollection".

Also by way of example, for an editability element of:

```
<xsf:xmlToEdit name="author" item="issue/@author "
    container="issue">
```

```
<xsf:editWith component="xOptional">
  <xsf:fragmentToInsert>
    <xsf:chooseFragment>
      <xsf:attributeData attribute="author" value="author
       name"/>
    </xsf:chooseFragment>
  </xsf:fragmentToInsert>
</xsf:editWith>
</xsf:xmlToEdit>
```

a node or nodes located with the XPath expression of "item/@author" has a permitted operation associated with the syntax "xOptional". This operation element comprises a character string of "editWith" and "component". The syntax following "component" (i.e., "xOptional") can be used by the editor application 130 to determine operations permitted for the editable node(s) indicated by the XPath expression "item/@author".

Like above, this operation, however, is conditional based on a context element (here an attribute of the editability element). This context is given by the value of the "container" attribute, in this case: "issue".

For the example immediately above, a syntax "fragmentToInsert" can be used by the editor application 130 to determine where and what to insert. Thus, the character string of "chooseFragment" indicates that an "author" attribute node having an value: "author name" is permitted to be added to the "issue" node.

Unlike the operation element having a syntax of "xCollection", the operation element corresponding to an "editWith" element whose "component"

attribute value has the value "xOptional", permits addition of only one node. It also permits that node (locatable with "item/@author") to then be deleted.

In another embodiment of the operation element, an attribute of the operation element indicates textual operations permitted on data of the editable node. This indication corresponds to an "editWith" element whose "component" attribute value has the value "xField", indicating that textual operations are permitted. Various types of textual operations may also be included in the electronic-form template 132.

For example, for an editability element of:

```
<xsf:xmlToEdit item="description/textItem">
    <xsf:editWith component="xField" type="rich" />
</xsf:xmlToEdit>
```

a node or nodes located with the XPath expression of "description/textItem" has a permitted operation specified by "editWith" and "component". A value of "component", here "xField", can be used to indicate that textual editing of data of the editable node is permitted. A value of "type", here "rich", can be used to indicate that rich-text editing is permitted. Thus, the syntax following "component" of "xField" and "rich" can be used by the editor application 130 to determine that rich-text editing of data of the editable node(s) indicated by the XPath expression "item/@author" is permitted.

In another embodiment of the "xField" operation element above, a type attribute given by a syntax "plain" indicates that an operation for creation and

modification of plain-text-data (but not rich-text-formatted data) of the editable node is permitted.

Fig. 13, for example, shows an electronic-form representation 1300 of an example of the XML document 134. Here many different nodes are presented and operations enabled. For instance, the electronic form 1300 shows data-entry fields enabling plain-text operations to be performed, referenced at numerals 1302 and 1304. It also shows enabling of other operations: a repeating table shown at numeral 1306; a bulleted list at 1308; and an optional section at 1310. Editability elements for these enabled operations can be represented in the electronic-form template 132 with:

```
<xsf:xmlToEdit name="term_115"
item="/po:Document/po:terms/po:term" >
 <xsf:editWith component="xTextList"/>
</xsf:xmlToEdit>
<xsf:editWith caption="notes" component="xOptional" >
 <xsf:fragmentToInsert>
  <xsf:chooseFragment>
   <po:notes/>
  </xsf:chooseFragment>
 </xsf:fragmentToInsert>
</xsf:editWith>
```

A Computer System

Fig. 14 shows an exemplary computer system that can be used to implement the processes described herein. Computer 1442 comprises one or more processors or processing units 1444, a system memory 1446, and a bus 1448 that couples various system components including the system memory 1446 to processors 1444. The bus 1448 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory 1446 comprises read only memory (ROM) 1450 and random access memory (RAM) 1452. A basic input/output system (BIOS) 1454, containing the basic routines that help to transfer information between elements within computer 1442, such as during start-up, is stored in ROM 1450.

Computer 1442 further comprises a hard disk drive 1456 for reading from and writing to a hard disk (not shown), a magnetic disk drive 1458 for reading from and writing to a removable magnetic disk 1460, and an optical disk drive 1462 for reading from or writing to a removable optical disk 1464 such as a CD ROM or other optical media. The hard disk drive 1456, magnetic disk drive 1458, and optical disk drive 1462 are connected to the bus 1448 by an SCSI interface 1466 or some other appropriate interface. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for computer 1442. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 1460 and a removable optical disk 1464, it should be appreciated by those skilled in the art that other types of computer-readable media

which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk 1456, magnetic disk 1460, optical disk 1464, ROM 1450, or RAM 1452, including an operating system 1470, one or more application programs 1472 (such as the design application 126 and the editor application 130), other program modules 1474, and program data 1476. A user may enter commands and information into computer 1442 through input devices such as a keyboard 1478 and a pointing device 1480. Other input devices (not shown) may comprise a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit 1444 through an interface 1482 that is coupled to the bus 1448. A monitor 1484 or other type of display device is also connected to the bus 1448 via an interface, such as a video adapter 1486. In addition to the monitor, personal computers typically comprise other peripheral output devices (not shown) such as speakers and printers.

Computer 1442 commonly operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 1488. The remote computer 1488 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically comprises many or all of the elements described above relative to computer 1442. The logical connections depicted in Fig. 14 comprise a local area network (LAN) 1490 and a wide area network (WAN) 1492. Such networking environments are

commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, computer 1442 is connected to the local network through a network interface or adapter 1494. When used in a WAN networking environment, computer 1442 typically comprises a modem 1496 or other means for establishing communications over the wide area network 1492, such as the Internet. The modem 1496, which may be internal or external, is connected to the bus 1448 via a serial port interface 1468. In a networked environment, program modules depicted relative to the personal computer 1442, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Generally, the data processors of computer 1442 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The system described herein comprises these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the blocks described, in conjunction with a microprocessor or other data processor. The system described can also comprise the computer itself when programmed according to the methods and techniques described herein.

For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks,

although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

Conclusion

The above-described system and method enables an end user to edit an XML document is ways permitted by an electronic-form template. Although the system and method has been described in language specific to structural features and/or methodological acts, it is to be understood that the system and method defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the claimed system and method.